PDFTron PDF PageMaster™ User Manual

Version 4.x

LEGAL STATEMENT AND COPYRIGHT NOTICE

PDFTron PDF PageMaster™ User Manual
Part number: PDFTRON-4-PDFPageMasterCMD
Part number: PDFTRON-4-PDFPageMasterSDK
Last Updated: February 1, 2010

TABLE OF CONTENTS

1. Introduction

1.1 An Introduction to PDFTron PDF PageMaster

PDFTron's **PDF PageMaster** is an easy-to-use, stand-alone command-line application providing users with an efficient means of splitting, editing, merging and securing PDF documents.

For developers who are looking for a software development component to integrate into their application, PDFTron also offers **PDF PageMaster SDK**, an easy-to-use, yet powerful software component for embedding into client and server based applications. PDF PageMaster SDK is available as a plain 'C DLL' and can be easily accessed from any programming language (including C#, VB.NET, C/C++, Java, VB6, Perl, Python, Ruby, Delphi, etc). Section 5 of this user manual deals specifically with PDF PageMaster SDK.

Like other PDFTron products, PDF PageMaster does not rely on any other third-party software. PDF PageMaster can be used in server environments or as a batch conversion process. Please see http://www.pdftron.com/pagemaster for more information.

PDF PageMaster is based on **PDFNet SDK**, PDFTron's own core technology. PDFNet SDK is a comprehensive developer library for PDF creation, manipulation and rendering, offered on a wide range of platforms and programming environments. If you require functionality for integration in your own applications beyond that which is provided as part of PDF PageMaster SDK, please contact a PDFTron representative or visit http://www.pdftron.com/pdfnet for more information.

1.1.1 Key Functions

- Merge multiple PDF documents into one.
- Split PDF documents into several files.
- Break PDF document into pieces based on chapters, bookmarks or file size.
- Extract or move specific pages or page ranges into new PDF documents.
- Remove specific pages from existing PDF documents.
- Insert or append new pages to any PDF document.
- Rearrange pages within a PDF document.
- Apply security settings, permissions, and document metadata to output PDF files.
- Get the number of pages in a given PDF document.
- Simple to use batch mode for processing entire folders with PDF documents.
- Options to preserve bookmarks, links, markup annotations and forms in the output PDF.
- Support for large PDF documents containing any number of pages.
- Create linearized and web ready PDF files with smallest file size.
- Get and set PDF document metadata.
- Fast, reliable and suitable for server use.
- Support for all versions of the PDF format (PDF 1.0 to ISO32000).
- Support for 128-bit AES (Advanced Encryption Standard) encryption and Crypt filters.
- Files with broken cross reference tables are automatically repaired.

1.1.2 Common Use Case Scenarios

PDF PageMaster is ideally suited to legal, insurance, and accounting firms, governments, healthcare, as well as any many other market sectors that extensively rely on documents in the PDF format. Some common-use case scenarios include:

- Server-based, on-demand delivery of dynamically assembled PDF documents.
- Assembling PDF books, brochures and catalogues by merging a set of input PDF documents, while preserving bookmarks, page links and forms.

- Inserting a cover page or appending a legal notice to every PDF document in a given directory.

1.1.3 Operating Systems Supported

- Windows 7, 2008, Vista, XP, 2003, 2000, NT, 98
- Mac OSX
- Linux

1.1.4 System Requirements

- At least 20 MB of free disk space.
- Memory requirement is heavily dependent on the nature of the document(s) being processed.

1.2 About This Manual

This manual is intended as a guide to the installation and use of PDF PageMaster. It is intended for system administrators, programmers and other users who are familiar with PDF documents, graphic file manipulation and general computer processes.

- Section 1 introduces PDF PageMaster and describes the manual.
- Section 2 explains how to install and uninstall PDF PageMaster.
- Section 3 summarizes the command-line arguments available for PDF PageMaster.
- Section 4 covers the basic usage of PDF PageMaster, including examples.
- Section 5 covers use of PDF PageMaster SDK.
- Section 6 includes Frequently Asked Questions.
- Section 7 is where you will find all the support information you may require, such as how to report a problem with the software.

2. Installing and Uninstalling PDF PageMaster

2.1 PDF PageMaster Installation

Both the PDF PageMaster Command-line Application as well as PDF PageMaster SDK is supplied as a download from a distributor or directly from www.pdftron.com. The release is packaged as a .zip file ('pagemaster.zip' or 'pagemaster_sdk.zip'). To install the software, simply unzip the archive in the desired location while making sure to preserve the directory (folder) structure during the process.

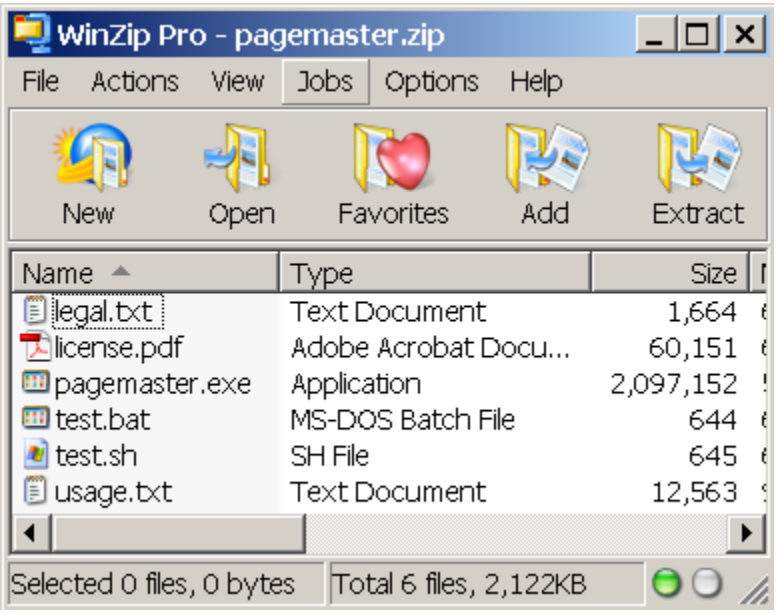

Figure 2.1
Extracting PDF PageMaster Archive using WinZip

2.2 Product Registration

After purchasing a license of PDF PageMaster, you will receive additional registration and license information via email.

If you purchased the PDF PageMaster Command-line Application, you just need to copy the license file provided to you into the "pagemaster" folder to register the software.

For PDF PageMaster SDK, you need to pass the license information during the call to PageMasterInit() to successfully register the SDK and remove evaluation restrictions. For example:

```
PageMasterInit("Joe Doe", "MyCompany", "sRLL4qr555sd63dnd");
```

The first parameter is your user name, the second parameter is your company name, and the last parameter is the license key supplied with your registration information.

2.3 Demo Version Installation

If you wish to evaluate the product, you can download the demo version of the product without any serial number or license key.

To do this, go to PDFTron's **Downloads** page at www.pdftron.com/downloads.html. Click on the appropriate product version/name, which will bring you to the product and the appropriate link for the demo download. Simply Download the zip file ('pagemaster.zip' or 'pagemaster_sdk.zip') and extract the archive in the desired location, while making sure to preserve the directory (folder) structure when extracting the archive. This will provide you with a working copy of the software. The limitation of the evaluation version is that all pages in processed documents will have a demo stamp.

2.4 Uninstalling PDF PageMaster

To remove PDF PageMaster from a computer, simply delete the "pagemaster" folder (for the command-line application) or the "pagemaster_sdk" folder (for PDF PageMaster SDK).

3. Overview

PDFTron's PDF PageMaster is an efficient solution designed for splitting, merging, editing, and securing PDF documents, while presenting several options to control output document settings. This section includes the command string syntax, used both in the PDF PageMaster Command-Line Application as well as in PDF PageMaster SDK.

3.1 Basic Syntax

The basic command-line syntax is:

```
pagemaster [options] file1 file2 folder1 file3 ...
```

3.2 Command-Line Summary

The following command-line arguments are available for PDF PageMaster.

Option	Parameter	Description
-o or --output	e.g. -o myfolder	The output folder used to store output files (for split, remove and insert). For merge, it specifies the output file.
--subfolders		Process all sub-directory for every directory specified in the argument list. By default, sub-directories are not processed.
-m or --merge		Merge documents, page ranges, or page sets.
--splice		This option is only relevant when merging pages. It can be used to assemble the new document sequence by interweaving pages from the given list of page sequences.
-s or --split		Split a PDF document.
--bybookmarks	--bybookmarks 1 or --bybookmarks 2,3	Split a PDF document into files based on bookmarks at the given depth or range of depths in the bookmark tree.
--byfilesize	--byfilesize 400	Split a PDF document into files of the given size in KB.
--digits	--digits 4	The number of digits used in the page number portion of the output filename for split operations. By default, new digits are added as needed; however this parameter could be used to format the page counter field to a uniform width (e.g. myfile_0001.pdf, myfile_0002.pdf, etc).
-r or --remove		Remove a given page or a list of pages from a PDF document.
-i or --insert	-i myfile.pdf --after 5	Insert a given file into every PDF document. For example, this option can be used to add cover pages to every PDF document in a given folder.
--before	-i cover.pdf --before 1	Specify that the file should be inserted before the given page.
--after	-i in.pdf --after 5	Specify that the file should be inserted after the given page.
--gettitle		Get the title of the document.
--getauthor		Get the name of the person who created the document.
--getsubject		Get the subject of the document.

--getkeywords		Get keywords associated with the document.
--getcreator		Get document's 'Creator' description.
--getproducer		Get document's 'Producer' description.
--getpagecount		Get the number of pages in the document.
--settitle	--settitle mytitle	Set document's 'Title' description.
--setauthor	--setauthor myauthor	Set document's 'Author' description.
--setsubject	--setsubject mysubject	Set document's 'Subject' description.
--setkeywords	--setkeywords mykeywords	Set document's 'Keywords' description.
--setcreator	--setcreator mycreator	Set document's 'Creator' description.
--setproducer	--setproducer myproducer	Set document's 'Producer' description.
--noprompt		Disable any user input. By default, the application will ask for a valid password if the password is incorrect.
--extension	--extension ".pdf"	The default file extension used to process PDF documents. The default is ".pdf".
-h or --help		Print a listing of available options.
-v or --version		Print the version information.
--verb	--verb 2	Set the verbosity level. Valid parameter values are 0-10. The higher number results in more feedback. The default is 1.
-d or --disable	e.g. to disable any changes to the document specify –d m e.g. to disable printing and content extraction specify –d phxdcm	This option accepts a string of permissions flags that should be disabled. The permission string may include the following flags: a - All permissions. p - Printing. m - Changing the document. c - Content copying or extraction. o - Commenting. f - Filling of form fields. x - Content extraction for accessibility. s - Document assembly. h – High quality printing.
-e or –enable	e.g. to enable commenting: -e o	A string of permissions flags (see --disable option above) that can be enabled. If there is a conflict with one of flags in the 'disable' option, the permission is not granted.
-p or --pass	-p open_pass	The password for secured PDF files. Not required if the input document(s) are not secured using the 'open' password.
-u or --userpass	-u result_pass	The new user password. The password required to open resulting documents.
-w or --ownerpass	-w owner_pass	The new owner/master password. This password is required to adjust permissions settings on resulting documents.
--AES	-u "My pass" --AES	Use 128-bit AES (Advanced Encryption Standard) Encryption.
--nobookmarks		Do not keep bookmarks when splitting or merging pages.
-l or --linearize		Linearize (web optimize) the output file(s).
--reverse		Reverse the page order of the output file(s).

4. Basic Usage and Examples

This section covers the basic usage of PDF PageMaster, explaining all the available options and illustrating examples.

4.1 Splitting PDF Documents

4.1.1 Split by Pages

You can split PDF documents based on manually defined page ranges with multitude of available options such as odd/even pages, reverse order, page ranges, page sets, etc.

Syntax: `pagemaster (-s or --split) <input>,<pagelist1>,<pagelist2>`

To split a PDF document by pages, specify -s (or –split) on the command-line followed by a list of one or more PDF documents to split.

For example, the following command-line will split ‘in.pdf’ and all PDF documents in ‘MyPDFs’ folder into individual PDF documents:

```
pagemaster -s in.pdf MyPDFs -o OutDir
```

The last option (-o OUT) instructs PageMaster to store all resulting documents in the new folder called ‘OutDir’.

The splitting process can be fine-tuned with a ‘page list’ parameter that can immediately follow the input PDF filename (after comma).

For example, the following command-line will extract the first five pages from ‘in.pdf’:

```
pagemaster -s in.pdf,1-5
```

For a detailed description of how to specify ‘page-lists’ as part of ‘Split-By-Page’ operation, please refer to Section 4.8 of this user manual. The following are some additional examples of how to use ‘page-lists’ to customize the splitting process:

Split ‘in.pdf’ into a set of PDF documents containing 2 pages each:

```
pagemaster -s in,2x
```

Split ‘in.pdf’ into two parts. The first PDF will contain pages 1 through 10, and the second document contains the remainder of the document:

```
pagemaster -s in,1-10,11-
```

The above command-line will break ‘in.pdf’ into several parts: A PDF containing pages 1 through 5, a PDF containing pages 6 to 10 (in the reverse order), and a set of odd pages in the page rage from page 15 to the last page in the document:

```
pagemaster -s in,1-5,10-6,15-odd
```

File Naming Convention for Split-By-Page Operation

Generally, by setting the '--digits' option, all numbers will be replaced by numbers with that number of digits: --digits <numdigits>

- If the user specifies a page range <firstpage>-<lastpage>(i.e. 1-5), then the created file will become <original file name>_<firstpage>-<lastpage> regardless of whether <firstpage> or <lastpage> are in the document.
- If - is specified, then the resulting filename is <original file name>_1-last.pdf
- If e or o is specified, then the resulting filename is <original file name>_<e or o>.pdf
- If <firstpage>-<lastpage><e or o> is specified, then the resulting filename is <original file name>_<firstpage>-<lastpage><e or o>
- If <# pages per doc>x is specified and <# pages per doc> is not 1, then each file created has the filename <original file name>_<start page>-(<start page> + <# pages per doc>)
  Even if the last document contains less than (<start page> + <# pages per doc>) pages
- If 1x is specified or no pagelist is specified (1x is used as the default), then each file created has the filename <original file name>_<page number>
- If <page number> is specified, then a file is created (with just that one page) with the filename <original file name>_<page number>

4.1.2 Split by Bookmarks

Most PDF utilities offer the ability to break documents by first or second level bookmarks only. With PDF PageMaster, you can split PDF documents by any bookmark level while retaining bookmarks and links in output files.

--bybookmarks <depth> will cause PDF PageMaster to split the pages of the given file based on the destinations of bookmarks that are at that depth in the bookmark tree. (Please note that pagelists are ignored if the '--bybookmarks' option is specified.)

--bybookmarks <depth1>,<depth2> will cause PDF PageMaster to split the pages of a given file based on the destinations of bookmarks that are at a depth between depth1 and depth2 in the bookmark tree.

Examples of Splitting by Bookmarks:

Split 'in.pdf' based on the destinations of the root bookmarks (usually chapters etc.):

```
pagemaster -s --bybookmarks 1 in.pdf
pagemaster -s --bybookmarks 1,1 in.pdf
```

Split 'in.pdf' based on root bookmark as well as the second level bookmark destinations:

```
pagemaster -s --bybookmarks 1,2 in.pdf
```

Split 'in.pdf' based on 3rd level bookmark destinations:

```
pagemaster -s --bybookmarks 3 in.pdf
```

File Naming Convention for Split-by-Bookmarks Operation

The file is named by the pages that it contains
<original file name>_<firstpage>-<lastpage> where firstpage is never larger than lastpage

4.1.3 Split by File Size

You can specify the size of your PDF files and PDF PageMaster will split the PDFs to the closest possible kilobyte.

--byfilesize <filesize> will cause PDF PageMaster to attempt to split with the file size of each created file being less than <filesize> KB. (It may not succeed in doing that if any one page is larger than <filesize> KB.)

Examples (assuming that each page can fit into the size provided):

Split largepdf.pdf into files of size 1000 KB or less:
```
pagemaster -s --byfilesize 1000 largepdf.pdf
```

Split largepdf.pdf into files of size 232 KB or less:
```
pagemaster -s --byfilesize 232 largepdf.pdf
```

File Naming Convention for Split-by-FileSize Operation
The file is named by the pages that it contains
<original file name>_<firstpage>-<lastpage> where firstpage is never larger than lastpage

4.2 Merging PDF Documents

PageMaster lets you to easily merge multiple PDF documents into a single output PDF document. You can select all pages or just a selected subset of pages (specific pages, page ranges, odd/even pages, etc) to be copied from each input document.

For a description of how to specify a pagelist as part of this operation, please refer to Section 4.8 of this user manual.

Syntax: `pagemaster (-m or --merge) <input1.PDF>,<pagelist1>,<pagelist2> <input2.PDF>,<pagelist3> <PDF folder> <input3.PDF> etc`.

To merge a list of PDF documents, specify -m (or --merge) on the command-line followed by a list of one or more PDF documents to split.

For example, the following command-line will merge ‘in1.pdf’, ‘in2.pdf’ and all PDF documents in ‘MyPDFs’ folder into a file called ‘merged.pdf’:

```
pagemaster –m in1.pdf in2.pdf MyPDFs -o Output/merged.pdf
```

The last option (-o merged.pdf) instructs PDF PageMaster to save the merged document in the new folder called ‘Output’.

The merging process can be fine-tuned with a ‘page list’ parameter that can immediately follow the input PDF filename (after comma). For example, the following command-line will merge the first five pages from ‘in.pdf’ and all PDF files stored in the ‘MyPDFs’ folder:

```
pagemaster -m in.pdf,1-5 MyPDFs,1-5
```

For a detailed description of how to specify ‘page-lists’ as part of the PDF merge operation, please refer to Section 4.8 of this user manual.

The following are some additional examples of how to use ‘page-lists’ to customize the merging process:

Merge all even pages from ‘in1.pdf’ with all odd pages from ‘in2.pdf’:
```
pagemaster -m in1.pdf,e in2.pdf,o -o out.pdf
```

Merge a PDF document by pulling pages 1 and 4 from ‘in1.pdf’, followed by a page range 5-10 (in the reverse order), followed by off pages in the rage 10-5 from ‘in2.pdf’, followed by all even pages from PDF files located under InDir1 folder:

```
pagemaster -m in1.pdf,1,4,10-5 in2.pdf,10-5o InDir1,e --subfolders -o merged.pdf
```

By default, ‘--merge’ will append the pages from the second file to the pages from the first file and so on. However, if the ‘--splice’ option is used, the merge function will interweave the documents. For example:

```
pagemaster -m in1.pdf in2.pdf in3.pdf -o out.pdf
```

will merge in1.pdf in2.pdf and in3.pdf to create a larger document out.pdf (in1.pdf followed by in2.pdf followed by in3.pdf)

```
pagemaster -m in1.pdf in2.pdf in3.pdf --splice -o out.pdf
```

In the above command-line, the document ‘out.pdf’ will be created starting with page 1 of ‘in1.pdf’. Then page 1 of in2.pdf and then page 1 of ‘in3.pdf’. It will continue in this manner until a document runs out of pages and then the other two will be interweaved in this manner until all pages are inserted.

4.3 Removing Pages

To remove specific pages or page ranges from PDF documents, use -r or --remove option.

Syntax: `pagemaster (-r or --remove) <input file 1>,<pagelist1>,<pagelist2>`

The operation will remove the given pages (in each page-list) from the input files. For a detailed description of how to specify ‘page-lists’ as part of PDF merge operation, please refer to Section 4.8 of this user manual. The following are some additional examples of how to use ‘page-lists’ to customize the merging process:

By default, the files will be overwritten, but the -o option can be used to specify an output directory. Please note that because PDF documents must contain at least one page, the ‘remove’ function cannot delete all pages in a document.

Examples:

Remove page 1 of firstpageremoved.pdf:
```
pagemaster -r firstpageremoved.pdf,1
```

Remove the first page from all PDF documents located in MyPDFs folder and its sub-folders:
```
pagemaster -r MyPDFs,1 --subfolders
```

Similar to above line, but save all files in the ‘Output’ folder instead of overwriting the originals:
```
pagemaster -r MyPDFs,1 --subfolders
```

Remove all odd pages from ‘in1.pdf’ and all even pages from ‘in2.pdf’:
```
pagemaster -r in1.pdf,o in2.pdf,e
```

Remove all even pages as well as the odd pages between 1 and 10 (1, 3, 5, 7, 9):
```
pagemaster -r mixedpages.pdf,e,1-10
```

4.4 Inserting/Appending a Document

Pages from multiple input documents can be sequentially inserted (or appended) into a new output document. Users can optionally specify a range of pages to use from each input document.

Syntax: `pagemaster (-i or --insert) <file to insert> <input files>`

By default, the inserted file is appended to the input files, but otherwise the following options can be used:

`--before <page number>`
causes the inserted file to be inserted before page <page number>

`--after <page number>`
causes the inserted file to be inserted after page <page number>

Examples of Inserting Files:

Insert toinsert.pdf before page 4 in inserted.pdf:
```
pagemaster -i toinsert.pdf --before 4 inserted.pdf
```

Append toinsert.pdf to append.pdf:
```
pagemaster -i toinsert.pdf append.pdf
```

Prepend toinsert.pdf to prepend.pdf:
```
pagemaster -i toinsert.pdf --before 1 prepend.pdf
```

4.5 Processing Folders

If the user specifies a folder rather than a specific PDF document, then all the PDF files inside of the folder will be processed in the specified way.

```
pagemaster <directory>|<password>,<page list 1>
```

that is, each file will be processed as if <directory> was the path of that file.

So, if directory pdf_files contained files pdf1.pdf pdf2.pdf and pdf3.pdf
pdf_files|pass,1-10 would be equivalent to pdf1.pdf|pass,1-10 pdf2.pdf|pass,1-10 pdf3.pdf|pass,1-10

PDF PageMaster also allows processing of subfolders, using the ‘--subfolders’ option.

If ‘--subfolders’ option is specified, the ‘remove’, ‘insert’ and ‘split’ functions will create the required subfolder directory structure at the output path, if required. (A directory is required if it contains PDF files or contains folders that are required.) The ‘merge’ function will merge all PDF files found in the subfolders into one PDF at the output location.

Examples:

The examples provided below are based on the following directory and file structure:

- directory pdf_dir contains files p1.pdf p2.pdf and p3.pdf as well as directories pdf_First and pdf_Second
- directory pdf_First contains directories pdf_Third and pdf_Fourth
- directory pdf_Second contains files p4.pdf and p5.pdf
- directory pdf_Third contains file p6.pdf
- directory pdf_Fourth contains only a text file ignored.txt

Merge files p1.pdf p2.pdf and p3.pdf and store the result in output.pdf:

```
pagemaster -m pdf_dir/ -o output.pdf
```

Merge files p1.pdf p2.pdf p3.pdf p4.pdf p5.pdf and p6.pdf and store the result in output.pdf:

```
pagemaster -m --subfolders pdf_dir/ -o output.pdf
```

Split files p1.pdf p2.pdf p3.pdf and store the resulting files p1_1-5.pdf p2_1-5.pdf and p3_1-5.pdf in output/

```
pagemaster -s pdf_dir/,1-5 -o output/
```

Create the directories output/pdf_First output/pdf_Second and output/pdf_First/pdf_Third, split the files p1.pdf p2.pdf p3.pdf p4.pdf p5.pdf and p6.pdf and store the resulting files p1_1-5.pdf p2_1-5.pdf and p3_1-5.pdf in output/, p4_1-5.pdf and p5_1-5.pdf in output/pdf_Second/ and p6_1-5 in output/pdf_First/pdf_Third/ (pdf_Fourth is not created since it is not required):

```
pagemaster -s --subfolders pdf_dir/,1-5 -o output/
```

Remove the first page of p1.pdf, p2.pdf and p3.pdf:

```
pagemaster -r pdf_dir/,1
```

Remove the first page of p1.pdf, p2.pdf, p3.pdf, p4.pdf, p5.pdf and p6.pdf:

```
pagemaster -r --subfolders pdf_dir/,1
```

Append insert.pdf to p1.pdf, p2.pdf and p3.pdf:

```
pagemaster -i insert.pdf pdf_dir/
```

Append insert.pdf to p1.pdf, p2.pdf, p3.pdf, p4.pdf, p5.pdf and p6.pdf:

```
pagemaster -i insert.pdf --subfolders pdf_dir/
```

4.6 Output Documents Options

PDF PageMaster can linearize output PDF documents for fast web or network access. This can greatly improve document loading time if you plan to share documents on the web or local network. To linearize (i.e web-optimize) all output documents add -l or --linearize option in the command-line.

Another option that controls the order of pages in the output PDF files is '--reverse' option. Adding '--reverse' in the command-line will reverse the page order in the output file(s).

Example:

```
pagemaster -s in.pdf,1-2 --linearize --reverse
```

Because of the '--reverse' command, the output file 'in_1-2.pdf' would contain page 2 (from 'in.pdf') followed by page 1. Also, because of the '--linearize' command, the resulting file will be web-optimized.

4.7 Security Options

PDF PageMaster presents several security options, such as encryption of documents with a password or options for allowing editing, printing, etc. You can control level of access to the document by restricting certain operations that can be performed on the files. Document editing and printing permissions can be also password protected.

4.7.1 Encryption

If a document is encrypted and the user does not wish to (or is not able to) enter the password on the command line, then the password can be specified to PDF PageMaster in the following ways:

1. Use the option -p (-p <password>) to specify a global password (PDF PageMaster will try to open all encrypted files with this password)

2. Specify a password for that particular file using the syntax <filename>|<password>,<page list 1>,... (This also works for directories if all files that are encrypted in that directory have the same password.)

To encrypt the output documents specify the -u option
-u <password for created files>

The default encryption algorithm is 128-bit RC4 encryption. However the '--AES' option can be used to change the encryption algorithm to 128-bit AES (Advanced Encryption Standard);

Examples:

The following commandline will attempt to decode file 'password_p1.pdf' using the password p1 and then merge its first five pages with otherfile.pdf:

```
pagemaster -m password_p1.pdf|p1,1-5 otherfile.pdf -o out.pdf
```

The following command-line will attempt to use the password 'pass' to decode any encrypted files in '1password_pass.pdf', '2password_pass.pdf', and '3password_pass.pdf'. The three files will then be merged:

```
pagemaster -m 1password_pass.pdf 2password_pass.pdf 3password_pass.pdf -p
pass -o out.pdf
```

The following command-line will attempt to use the password ‘pass’ to decode any encrypted files in ‘2password_pass.pdf’ and ‘3password_pass.pdf’ and ‘pw’ to decode ‘1password_pw.pdf’ (and then try pass if pw is not the password of 1password_pw.pdf). The three files will then be merged.

```
pagemaster -m 1password_pw|pw 2password_pass 3password_pass -p pass –o
out.pdf
```

Create files file_2-3.pdf with pages 2-3 of file.pdf and file_11.pdf with page 11 of file.pdf.
Both of these documents will then be encrypted with 128-bit RC4 encryption and the password out_pass:

```
pagemaster -s -u out_pass file.pdf,2-3,11
```

Create files: file_2-5.pdf with pages 2-5 of file.pdf and file_7.pdf with page 7 of file.pdf.
Both of these documents will then be encrypted with 128-bit AES encryption and the password out_pass:

```
pagemaster -s -u --AES out_pass file.pdf,2-5,7
```

4.7.2 Permissions

To set the permissions password (which is used to prevent permissions from being changed), specify the -w option: -w <master password>

To set the permissions, use the options ‘--enable’ and ‘--disable' with the following values:

Value	Description
a	All permissions.
p	Printing.
m	Changing the document.
c	Content copying or extraction.
o	Commenting.
f	Filling of form fields.
x	Content extraction for accessibility.
s	Document assembly.
h	High quality printing.

Examples:

Merge m1.pdf and m2.pdf to create result.pdf and set its permissions so that printing, document assembly, and high quality printing are allowed, but Content copying or extraction, Commenting and Filling of form fields are not allowed:

```
pagemaster -m m1.pdf m2.pdf --enable  psh --disable cof -o result.pdf
```

Merge m1.pdf and m2.pdf to create result.pdf and set its permissions so that printing, document assembly, and high quality printing are allowed, but Content copying or extraction, Commenting and Filling of form fields are not allowed:

```
pagemaster -m m1.pdf m2.pdf --enable  a --disable cof -o result.pdf
```

4.8 Specifying Pagelists for Merge, Split and Remove Functions.

PDF PageMaster includes a variety of options for selecting specific pages or page ranges while performing the basic operations.

Pagelists:

1. e, o gets even and odd pages in the document respectively. This will allow you to merge, split or remove by even/odd pages.
2. - gets all pages in the document
3. <start page number>- gets page <start page number> and all subsequent pages
4. -<end page number> gets page <end page number> and all previous pages
5. <page number> gets the specific page <page number>
6. <start page number>-<end page number> gets all pages between and including <start page number> and <end page number>
If <start page number> is greater than <end page number>, then the pages are inserted in reverse order.
7. Rules 3 and 5 can be combined to get all even or odd pages between <start page number> and <end page number>
e.g. <start page number>-<end page number><e or o>

Split-Specific Rules:

1. The syntax <pages per doc>x can be used to split the document into files with <pages per doc> pages each.
2. With the exception of the above rule, one file will be created for every page list.

Examples:

Produce the file out.pdf containing the following attributes:
1. In ascending order the even pages of ‘input.pdf’
2. The first page of ‘input.pdf’
3. All pages of ‘extra.pdf’ in order

```
pagemaster -m input.pdf,e,1 extra.pdf -o out.pdf
```

4.9 Batch Processing and the Use of Wildcards

PDF PageMaster supports processing of multiple input documents in the same run. For example, it is possible to specify multiple PDF folders and PDF PageMaster will automatically process all PDF documents matching a given file extension. For example, the following command-line will process all PDF documents in folders ‘test1’ and ‘test2’

```
c:\>pagemaster -o c:/output_folder c:/test1 c:/test2
```

Wildcard characters can also be used to process multiple input files.

For example, if a directory contains the following PDF documents:

```
C:\test1 >dir
 Directory of C:\test1
01/04/2007  03:35 PM    <DIR>          .
01/04/2007  03:35 PM    <DIR>          ..
05/21/2004  02:27 PM             A1.pdf
05/03/2005  09:38 AM             A2.pdf
05/20/2003  08:46 AM             B1.pdf
05/15/2003  12:50 PM             B2.pdf
```

To merge all PDF documents in this folder, you could specify:

```
c:\>pagemaster -m -o c:/output_file.pdf c:/test1/*.pdf
```

To merge all PDF documents staring with ‘A’, you could specify:

```
pagemaster -o c:/output_file.pdf c:/test1/A*.pdf
```

Or, to process all PDF documents ending with ‘1’, you could specify:

```
pagemaster -o c:/output_file.pdf c:/test1/*1.pdf
```

You can use either of the two standard wildcards — the question mark (?) and the asterisk (*) — to specify filename and path arguments on the command-line.

The wildcards are expanded in the same manner as operating system commands. (Please refer to your operating system user's guide if you are unfamiliar with wildcards). Enclosing an argument in double quotation marks (" ") suppresses the wildcard expansion. Within quoted arguments, you can represent quotation marks literally by preceding the double-quotation-mark character with a backslash (****). If no matches are found for the wildcard argument, the argument is passed literally.

4.10 Exit Codes

To provide additional feedback, PDF PageMaster returns exit codes after completing processing. The exit codes can be used to provide user feedback, for logging etc. This is particularly important for applications running in an unattended environment.

The following table lists possible exit codes and their description:

Exit Code	Description
0	No errors. All files converted successfully.
1	Unspecified error.
2	Bad license key.
3	Failed to create the output directory.
4	Bad input filename or path.
5	Output file not specified.
6	Error reading the input document.
7	Document is secured. Need a valid password to open the document.
8	Error writing an output file or folder
9	No input files found

All codes other then ‘0’ indicate that there was an error during the conversion process.

The following illustrates a sample Windows batch script that processes exit codes:

```
@echo off rem run a simple merge operation...
pagemaster -o OUT -m in1.pdf in2.pdf In1Folder
if errorlevel 9 goto inputerr
if errorlevel 7 goto passwd
if errorlevel 1 goto othererror
if errorlevel 0 goto exit

:passwd
echo Document is protected. Need a valid password to open the document.
goto exit

:inputerr
echo No input files specified.
goto exit

:othererror
echo An error encountered during processing.
goto exit

:exit
```

5 Overview of PDF PageMaster SDK

PDFTron's PageMaster SDK offers a simple-to-use API for splitting and merging PDF documents. The SDK is available as a plain 'C DLL' and can be easily accessed from any programming language (including C#, VB.NET, C/C++, Java, VB6, Perl, Python, Ruby, Delphi, etc).

The entire API consists of only two functions and is very easy to use.

This section covers the basic use of PageMaster SDK explaining all the available options.

PDF PageMaster is based on **PDFNet SDK**, PDFTron's own core technology. PDFNet SDK is a comprehensive developer library for PDF creation, manipulation and rendering, offered on a wide range of platforms and programming environments. If you require functionality for integration in your own applications beyond that which is provided as part of PDF PageMaster SDK, please contact a PDFTron representative or visit http://www.pdftron.com/net for more information.

5.1 Working with the PDF PageMaster API

PDF PageMaster API consists of only two functions: PageMasterInit and PageMasterRun.

PageMasterInit is called only once per process session to initialize the library and register the component. PageMasterRun can be called many times to process PDF documents or folders with PDF documents.

The following is the simplest application that can be built using PageMaster SDK:

```
// Using C# or C/C++
void main() {
  PageMasterInit("username", "company", "lic_key");
  PageMasterRun("-s -o OUTDIR in.pdf", null, new IntPtr(0));
}
```

This application essentially executes a hardcoded operation that splits "in.pdf" into a set of PDF documents (one output PDF for every page in the input PDF document). The resulting PDF files will be stored in the "OUTDIR" folder.

The first parameter of the PageMasterRun() function is a command string which is exactly the same as the general syntax used for the PageMaster Command-Line application. For a detailed explanation of all options, please refer to section 4 of this manual. The PageMaster Command-Line application is a great tool to get to know all the available options. In fact, building a command-line application using PageMaster SDK is as simple as the following listing:

```
// Using C#
static void Main(string[] args) {

  PageMasterInit("username", "company", "lic_key");

  String s = "";
  foreach (string arg in args) {
    s += arg + " ";
  }
  PageMasterRun(s, null, new IntPtr(0));
}
```

It is also possible to build the command string dynamically (e.g. based on the user or dynamic input), as illustrated in the following code snippet:

```
// Using C#
static void Main(string[] args)
{
	PageMasterInit("username", "company", "lic_key");

	bool linearized_output = true;
	bool aes_encryption = false;
	string output_file = "TestOutput/test3/test.pdf";
	string open_password = "mypass";
	string title = "My Title";
	string author = "Joe Doe";
	string subject = "My Subject";
	string keywords = "key1 key2 key2";
	string creator = "PDFTron PDFNet";
	string producer = "PageMaster";

	// Permissions
	bool printing = false;
	bool high_rez_printing = false;
	bool doc_modifications = true;  // Changing the document
	bool content_extraction = true;
	bool commenting = true;
	bool forms_editing = true;
	bool accessibility = true;
	bool document_assembly = true;

	// Given the above settings build a command string.
	string s = "";

	//specify the merge option
	s += "-m ";

	if (!(output_file=="")) s += "-o " + output_file + " ";

	if (!(open_password=="")) s += "--userpass " + open_password + " ";
	if(aes_encryption) s+= "--AES ";


	// Set the permissions.
	s += "-e a -d \"";
	if (printing == false) s += "p";
	if (high_rez_printing == false) s += "h";
	if (doc_modifications == false) s += "m";
	if (content_extraction == false)  s += "c";
	if (commenting == false)  s += "o";
	if (forms_editing == false)  s += "f";
	if (accessibility == false)  s += "x";
	if (document_assembly == false)  s += "s";
	s += "\" ";

	if (linearized_output) s += "-l ";

	// Set document information.
```

```
        if (!(title==""))  s += "--settitle \"" + title + "\" ";
        if (!(author==""))  s += "--setauthor \"" + author + "\" ";
        if (!(subject==""))  s += "--setsubject \"" + subject + "\" ";
        if (!(keywords==""))  s += "--setkeywords \"" + keywords + "\" ";
        if (!(creator==""))  s += "--setcreator \"" + creator + "\" ";
        if (!(producer==="")) s += "--setproducer \"" + producer + "\" ";

        // specify input files and folders...
        s += "TestFiles/Bookmark.pdf,1-15 ";
        s += "TestFiles/secured.pdf|test ";
        s += "--verb 10";

        //an arraylist to store the saved file path(s)
        ArrayList saved_files=new ArrayList();

        //create a GCHandle to allow MyCallback to access
        //the object saved_files
        GCHandle gch = GCHandle.Alloc(saved_files);
        // Execute the command string.
        PageMasterRun(s, mycallback, (IntPtr)gch );
        gch.Free();

        //print the list of output file(s)
        Console.WriteLine("The Saved File Is:");
        for(int i=0; i<saved_files.Count; i++)
        {
              Console.WriteLine(saved_files[i]);
        }
}
```

5.2 Reporting Progress Messages and Errors

To find out if PageMaster processing was successful, the application can query the status code returned by PageMasterRun().

For example,

```
int ret = PageMasterRun(...);
if (ret == PAGEMASTER_OK) {
      // No errors...
}
else if (ret == PAGEMASTER_ERR_BADKEY ) {
      // bad license key...
}
else if (ret == PAGEMASTER_ERR_DIRCREATE) {
      // Failed to create the output directory
}
else {
      // Other error
}
```

A non-zero value indicates that an error was encountered. You can find the listing for all error codes in 'include/PageMaster.h' header.

For more detailed error and message reporting, you can pass a pointer to the custom callback function in the second parameter of PageMasterRun(). The last parameter in PageMasterRun is a pointer to custom data that you may want to pass to the callback function.

A sample callback function may look as follows:

```
// Using C/C++
char* MyCallback(int mode, char* msg, void* user_data) {
  if (mode == PAGEMASTER_ERROR) {
    cout << "Error: " << msg << endl;
  }
  else if (mode == PAGEMASTER_MSG) {
    cout << msg;
  }
  else if (mode == PAGEMASTER_GETPASS) {
    static string gl_pass;
    cin >> gl_pass;
    return (char*)gl_pass.c_str();
  }
  return 0;
}

' or in VB.NET...
Public Function MyCallback(ByVal mode As Integer, ByVal msg As String,
ByVal user_data As Int32) As Int32
  If mode = PAGEMASTER_ERROR Then
   Console.WriteLine("Error: {0}", msg)
  ElseIf mode = PAGEMASTER_MSG Then
   Console.Write("{0}", msg)
  End If
  Return 0
End Function    'MyCallback
```

6 Frequently Asked Questions

6.1 Does PDF PageMaster have any dependencies on third party components/software?

PDF PageMaster is a completely stand-alone application and it does not include any dependencies on third-party components or software.

6.2 What is the difference between the PDF PageMaster Command-line Application and PDF PageMaster SDK?

PageMaster SDK and the Command-Line Application offer exactly the same PDF page manipulation functionality. The main difference between the two software editions is how the functionality is packaged and accessed. The PageMaster Command-Line Application can be invoked from the command-line, batch scripts, or via a shell command. PageMaster SDK, on the other hand, is primarily intended for integration with third-party applications; it includes a DLL (Dynamic Linked Library) and a set of sample projects illustrating how to use the DLL from C#, VB, Java, and C/C++. One of the advantages of the SDK edition is that it is possible to run multiple threads of execution within the same process.

6.3 What is the difference between PDFNet SDK and PDF PageMaster SDK?

PDF PageMaster is based on **PDFNet SDK**, PDFTron's own core PDF programming technology. PDFNet SDK is a comprehensive developer library for PDF creation, manipulation and rendering, offered on a wide range of platforms and programming environments. If you require functionality for integration in your own applications beyond that which is provided as part of PDF PageMaster SDK, or need more flexibility in implementing the required functionality, please contact a PDFTron representative or visit http://www.pdftron.com/pdfnet for more information.

The main distinction between the two components is that PageMaster SDK is a specialized tool for simple PDF manipulations such as page splitting and merging, whereas PDFNet SDK is a general purpose Software Development Toolkit (SDK) that can be used to implement any type of manipulation on PDF documents (including splitting and merging) as well as other type of PDF functionality.

The advantage of PDF PageMaster is that it is very simple to use and that it doesn't require much experience with PDF format or general programming. At the same time PDF PageMaster does not include all of the features and options that are available in PDFNet SDK (http://www.pdftron.com/pdfnet).

6.4 Can I integrate PDF PageMaster SDK with my client/server application?

PDF PageMaster has a simple-to-use API that can be easily integrated into third-party client and server-based applications. The SDK is available as a plain 'C DLL' and can be easily accessed from any programming language (including C#, VB.NET, C/C++, Java, VB6, Perl, Python, Ruby, Delphi, etc). For more information on PDF PageMaster SDK, please contact a PDFTron representative at info@pdftron.com.

6.5 I didn't find the answer to my question in the user manual. Are there any other helpful resources?

You may want to search the PDF PageMaster Knowledge Base forum (which can be accessed via PDFTron's website at: www.pdftron.com) or to simply forward your question to PDFTron's technical support team via support@pdftron.com.

7 Support

7.1 Reporting Problems

If you encounter a problem or question regarding PDFTron PDF PageMaster, which is not addressed on PDFTron's website, please submit a problem report to PDFTron's Support group at http://www.pdftron.com/reportproblem.html.

When submitting a problem you will be asked to provide the following information:

- Contact details
- Product and Version of the product
- Detailed description of problem
- Problem file(s)
- Whether you have an AMS (Annual Maintenance Subscription).
- Any other information that may be related

7.2 Contact Information

To contact PDFTron directly, please use the contact information below:

Tel: 1-604-730-8989
Fax: 1-604-676-2477

Web site: www.pdftron.com

Email Contacts:

General Business Inquiries: info@pdftron.com
Sales & Licensing: sales@pdftron.com
Product Support: support@pdftron.com
Professional Services: services@pdftron.com
Website related questions: webmaster@pdftron.com
Press & News: press@pdftron.com